EXHIBIT 8

This paper was released Sept. 13, 2006 and published in *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop* (EVT'07). For an extended version of this paper and videos of demonstration attacks, see http://citp.princeton.edu/voting.

Security Analysis of the Diebold AccuVote-TS Voting Machine

Ariel J. Feldman[*], J. Alex Halderman[*], and Edward W. Felten[*,†]

[*]Center for Information Technology Policy and Dept. of Computer Science, Princeton University
[†]Woodrow Wilson School of Public and International Affairs, Princeton University
{ajfeldma,jhalderm,felten}@cs.princeton.edu

Abstract

This paper presents a fully independent security study of a Diebold AccuVote-TS voting machine, including its hardware and software. We obtained the machine from a private party. Analysis of the machine, in light of real election procedures, shows that it is vulnerable to extremely serious attacks. For example, an attacker who gets physical access to a machine or its removable memory card for as little as one minute could install malicious code; malicious code on a machine could steal votes undetectably, modifying all records, logs, and counters to be consistent with the fraudulent vote count it creates. An attacker could also create malicious code that spreads automatically and silently from machine to machine during normal election activities—a voting-machine virus. We have constructed working demonstrations of these attacks in our lab. Mitigating these threats will require changes to the voting machine's hardware and software and the adoption of more rigorous election procedures.

1 Introduction

The Diebold AccuVote-TS and its newer relative the AccuVote-TSx are together the most widely deployed electronic voting platform in the United States. In the November 2006 general election, these machines were used in 385 counties representing over 10% of registered voters [12]. The majority of these counties—including all of Maryland and Georgia—employed the AccuVote-TS model. More than 33,000 of the TS machines are in service nationwide [11].

This paper reports on our study of an AccuVote-TS, which we obtained from a private party. We analyzed the machine's hardware and software, performed experiments on it, and considered whether real election practices would leave it suitably secure. We found that the machine is vulnerable to a number of extremely serious attacks that undermine the accuracy and credibility of the vote counts it produces.

Figure 1: The Diebold AccuVote-TS voting machine

Computer scientists have been skeptical of voting systems of this type, Direct Recording Electronic (DRE), which are essentially general-purpose computers running specialized election software. Experience with computer systems of all kinds shows that it is exceedingly difficult to ensure the reliability and security of complex software or to detect and diagnose problems when they do occur. Yet DREs rely fundamentally on the correct and secure operation of complex software programs. Simply put, many computer scientists doubt that paperless DREs can be made reliable and secure, and they expect that any failures of such systems would likely go undetected.

Previous security studies of DREs affirm this skepticism (e.g., [7, 18, 22, 30, 39]). Kohno, Stubblefield, Rubin, and Wallach studied a leaked version of the source code for parts of the Diebold AccuVote-TS software and found many design errors and vulnerabilities [22]. Hursti later examined the hardware and compiled firmware of AccuVote-TS and TSx systems and discovered problems with the software update mechanism that could allow malicious parties to replace the programs that operate the

machines [18]. Our study confirms these results by building working demonstrations of several previously reported attacks, and it extends them by describing a variety of serious new vulnerabilities.

Main Findings The main findings of our study are:

1. Malicious software running on a single voting machine can steal votes with little risk of detection. The malicious software can modify all of the records, audit logs, and counters kept by the voting machine, so that even careful forensic examination of these records will find nothing amiss. We have constructed demonstration software that carries out this vote-stealing attack.

2. Anyone who has physical access to a voting machine, or to a memory card that will later be inserted into a machine, can install said malicious software using a simple method that takes as little as one minute. In practice, poll workers and others often have unsupervised access to the machines.

3. AccuVote-TS machines are susceptible to voting-machine viruses—computer viruses that can spread malicious software automatically and invisibly from machine to machine during normal pre- and post-election activity. We have constructed a demonstration virus that spreads in this way, installing our demonstration vote-stealing program on every machine it infects. Our demonstration virus spreads via the memory cards that poll workers use to transfer ballots and election results, so it propagates even if the machines are not networked.

4. While some of these problems can be eliminated by improving Diebold's software, others cannot be remedied without replacing the machines' hardware. Changes to election procedures would also be required to ensure security.

The details of our analysis appear below, in the main body of this paper.

Given our findings, we believe urgent action is needed to address these problems. We discuss potential mitigation strategies below in Section 5.

The machine we obtained came loaded with version 4.3.15 of the Diebold BallotStation software that runs the machine during an election.[1] This version was deployed in 2002 and certified by the National Association of State Election Directors (NASED) [15]. While some of the problems we identify in this report may have been remedied in subsequent software releases (current versions are in the 4.6 series), others are architectural in nature and cannot easily be repaired by software changes. In any case, subsequent versions of the software should be assumed insecure until fully independent examination proves otherwise.

Though we studied a specific voting technology, we expect that a similar study of another DRE system, whether from Diebold or another vendor, would raise similar concerns about malicious code injection attacks and other problems. We studied the Diebold system because we had access to it, not because it is necessarily less secure than competing DREs. All DREs face fundamental security challenges that are not easily overcome.

Despite these problems, we believe that it is possible, at reasonable cost, to build a DRE-based voting *system*—including hardware, software, and election procedures—that is suitably secure and reliable. Such a system would require not only a voting machine designed with more care and attention to security, but also an array of safeguards, including a well-designed voter-verifiable paper audit trail system, random audits and forensic analyses, and truly independent security review.[2]

Outline The remainder of this paper is structured as follows. Section 2 describes several classes of attacks against the AccuVote-TS machine as well as routes for injecting malicious code. Section 3 discusses the machine's design and its operation in a typical election, focusing on design mistakes that make attacks possible. Section 4 details our implementation of demonstration attacks that illustrate the security problems. Section 5 examines the feasibility of several strategies for mitigating all of these problems. Section 6 outlines prior research on the AccuVote system and DREs more generally. Finally, Section 7 offers concluding remarks.

2 Attack Scenarios

Elections that rely on Diebold DREs like the one we studied are vulnerable to several serious attacks. Many of these vulnerabilities arise because the machine does not even attempt to verify the authenticity of the code it executes. In this section we describe two classes of attacks—vote stealing and denial-of-service [20]—that involve injecting malicious code into the voting machine. We then outline several methods by which code can be injected and discuss the difficulty of removing malicious code after a suspected attack.

[1] The behavior of our machine conformed almost exactly to the behavior specified by the source code to BallotStation version 4.3.1, which leaked to the public in 2003.

[2] Current testing agencies are often referred to as "independent testing agencies" (ITAs), but "independent" is a misnomer, as they are paid by and report to the voting machine vendor.

2.1 Classes of Attacks

2.1.1 Vote-Stealing Attacks

The AccuVote-TS machine we studied is vulnerable to attacks that steal votes from one candidate and give them to another. Such attacks can be carried out without leaving any evidence of fraud in the system's logs. We have implemented a demonstration attack to prove that this is possible; it is described in Section 4.2.

To avoid detection, a vote-stealing attack must transfer votes from one candidate to another, leaving the total number of votes unchanged so that poll workers do not notice any discrepancy in the number of votes reported. Attacks that only add votes or only subtract votes would be detected when poll workers compared the total vote count to the number of voters who signed in at the desk.[3]

The machine we studied maintains two records of each vote—one in its internal flash memory and one on a removable memory card. These records are encrypted, but the encryption is not an effective barrier to a vote-stealing attack because the encryption key is stored in the voting machine's memory where malicious software can easily access it. Malicious software running on the machine would modify both redundant copies of the record for each vote it altered. Although the voting machine also keeps various logs and counters that record a history of the machine's use, a successful vote-stealing attack would modify these records so they were consistent with the fraudulent history that the attacker was constructing. In the Diebold DRE we studied, these records are stored in ordinary flash memory, so they are modifiable by malicious software.

Such malicious software can be grafted into the BallotStation election software (by modifying and recompiling BallotStation if the attacker has the BallotStation source code, or by modifying the BallotStation binary), it can be delivered as a separate program that runs at the same time as BallotStation, it can be grafted into the operating system or bootloader, or it can occupy a virtualized layer below the bootloader and operating system [21]. The machine contains no security mechanisms that would detect a well designed attack using any of these methods. However it is packaged, the attack software can modify each vote as it is cast, or it can wait and rewrite the machine's records later, as long as the modifications are made before the election is completed.

The attack code might be constructed to modify the machine's state only when the machine is in election mode and avoid modifying the state when the machine is performing other functions such as pre-election logic and accuracy testing. The code could also be programmed to operate only on election days. (Elections are often held according to a well-known schedule—for example, U.S. presidential and congressional elections are held on the Tuesday following the first Monday of November, in even-numbered years.) Alternatively, it could be programmed to operate only on *certain* election days, or only at certain times of day.

By these methods, malicious code installed by an adversary could steal votes with little chance of being detected by election officials.[4] Vote counts would add up correctly, the total number of votes recorded on the machine would be correct, and the machine's logs and counters would be consistent with the results reported—but the results would be fraudulent.

2.1.2 Denial-of-Service Attacks

Denial-of-service (DoS) attacks aim to make voting machines unavailable on election day or to deny officials access to the vote tallies when the election ends [20, 28, 3]. It is often known in advance that voters at certain precincts, or at certain times, will vote disproportionately for one party or candidate. A targeted DoS attack can be designed to distort election results or to spoil an election that appears to be favoring one party or candidate. Several kinds of DoS attacks are practical on the AccuVote-TS system because of the ease with which malicious code may be executed.

One style of DoS attack would make voting machines unavailable on election day. For example, malicious code could be programmed to make the machine crash or malfunction at a pre-programmed time, perhaps only in certain polling places. In an extreme example, an attack could strike on election day, perhaps late in the day, and completely wipe out the state of the machine by erasing its flash memory. This would destroy all records of the election in progress, as well as the bootloader, operating system, and election software. The machine would refuse to boot or otherwise function. The machine would need to be serviced by a technician to return it to a working state. If many machines failed at once, available technicians would be overwhelmed. Even if the machines were repaired, all records of the current election would be lost. (We have created a demonstration version of this attack, which is described below in Section 4.4.) A similar style of DoS attack would try to spoil an election by modifying the machine's vote counts or logs in a manner that would be easy to detect but impossible to correct, such as adding or removing so many votes that the resulting totals would

[3] It might be possible to subtract a few votes without detection (if poll workers interpret the missing votes as voters who did not vote in that race) or to add a few votes to compensate for real voters who did not cast ballots; but in any case transferring votes from one candidate to another is a more effective attack.

[4] Officials might try to detect such an attack by parallel testing. As we describe in Section 5.3, an attacker has various countermeasures to limit the effectiveness of such testing.

be obviously wrong. A widespread DoS attack of either style could require the election to be redone.

2.2 Injecting Attack Code

To carry out these attacks, the attacker must somehow install his malicious software on one or more voting machines. If he can get physical access to a machine for as little as one minute, he can use attacks discovered by Hursti [18] to install the software manually. The attacker can also install a voting machine virus that spreads to other machines, allowing him to commit widespread fraud even if he only has physical access to one machine or memory card.

2.2.1 Direct Installation

An attacker with physical access to a machine would have at least three methods of installing malicious software. The first is to create an EPROM chip containing a program that will install the attack code into the machine's flash memory, and then to open the machine, install the chip on its motherboard, and reboot from the EPROM.[5]

The second method is to exploit a back door feature in Diebold's code, first discovered by Hursti. This method allows the attacker to manually install attack software from a memory card. When the machine boots, it checks whether a file named `explorer.glb` exists on the removable memory card. If such a file is present, the machine boots into Windows Explorer rather than Diebold's BallotStation election software. An attacker could insert a memory card containing this file, reboot the machine, and then use Explorer to copy the attack files onto the machine or run them directly from the card. [18]

The third method exploits a service feature of the machine's bootloader, also discovered by Hursti. On startup, the machine checks the removable memory card for a file named `fboot.nb0`. If this file exists, the machine replaces the bootloader code in its on-board flash memory with the file's contents. An attacker could program a malicious bootloader, store it on a memory card as `fboot.nb0`, and reboot the machine with this card inserted, causing the Diebold bootloader to install the malicious software [18]. (A similar method would create a malicious operating system image.)

The first method requires the attacker to remove several screws and lift off the top of the machine to get access to the motherboard and EPROM. The other methods only require access to the memory card slot and power button, which are both behind a locked door on the side of the machine.[6] The lock is easily picked—one member of our group, who has modest locksmithing skills, can pick the lock consistently in less than 10 seconds. Moreover, in their default configuration, all AccuVote-TS machines can be opened with the same key [4], and copies of this key are not difficult to obtain. The particular model of key that the AccuVote-TS uses is identified by an alphanumeric code printed on the key. A Web search for this code reveals that this exact key is used widely in office furniture, jukeboxes, and hotel mini bars, and is for sale at many online retailers. We purchased copies of the key from several sources and confirmed that they all can open the machine.

A poll worker, election official, technician, or other person who had private access to a machine for as little as one minute could use these methods with little risk of detection. Poll workers often do have such access; for instance, in a widespread practice called "sleepovers," machines are sent home with poll workers the night before the election [35].

2.2.2 Voting Machine Viruses

Rather than injecting code into each machine directly, an attacker could create a computer virus that would spread from one voting machine to another. Once installed on a single "seed" machine, the virus would spread to other machines by methods described below, allowing an attacker with physical access to one machine (or card) to infect a potentially large population of machines. The virus could be programmed to install malicious software, such as a vote-stealing program or denial-of-service attack, on every machine it infected.

To prove that this is possible, we constructed a demonstration virus that spreads itself automatically from machine to machine, installing our demonstration vote-stealing software on each infected system. Our demonstration virus, described in Section 4.3, can infect machines and memory cards. An infected machine will infect any memory card that is inserted into it. An infected memory card will infect any machine that is powered up or rebooted with the memory card inserted. Because cards are transferred between machines during vote counting and administrative activities, the infected population will grow over time.

Diebold delivers software upgrades to the machines via memory cards: a technician inserts a memory card containing the updated code and then reboots the machine, causing the machine's bootloader to install the new code from the memory card. This upgrade method relies on the correct functioning of the bootloader, which is supposed to copy the upgraded code from the memory card into the machine's flash memory. But if the bootloader is

[5] When the machine is rebooted, it normally emits a musical chime that might be noticed during a stealth attack; but this sound can be suppressed by plugging headphones (or just a headphone connector) into the machine's headphone jack.

[6] The locked door must be opened in order to remove one of the screws holding the machine's top on.

already infected by a virus, then the virus can make the bootloader behave differently. For example, the bootloader could pretend to install the updates as expected but instead secretly propagate the virus onto the memory card. If the technician later used the same memory card to "upgrade" other machines, he would in fact be installing the virus on them. Our demonstration virus illustrates these spreading techniques.

Memory cards are also transferred between machines in the process of transmitting election definition files to voting machines before an election. According to Diebold, "Data is downloaded onto the [memory] cards using a few [AccuVote] units, and then the stacks of [memory] cards are inserted into the thousands of [AccuVote] terminals to be sent to the polling places." ([10], p. 13) If one of the few units that download the data is infected, it will transfer the infection via the "stacks of [memory] cards" into many voting machines.

2.3 Difficulty of Recovery

If a voting machine has been infected with malicious code, or even if infection is suspected, it is necessary to disinfect the machine. The only safe way to do this is to put the machine back into a known-safe state, by, for example, overwriting all of its stable storage with a known configuration.

This is difficult to do reliably. We cannot depend on the normal method for installing firmware upgrades from memory cards, because this method relies on the correct functioning of the bootloader, which might have been tampered with by an attacker. There is no foolproof way to tell whether an update presented in this way really has been installed safely.

The only assured way to revert the machine to a safe state is to boot from EPROM using the procedure described in Section 3. This involves making an EPROM chip containing an update tool, inserting the EPROM chip into the motherboard, setting the machine to boot from the chip, and powering it on. On boot, the EPROM-based updater would overwrite the on-board flash memory, restoring the machine to a known state. Since this process involves the insertion (and later removal) of a chip, it would probably require a service technician to visit each machine.

If the disinfection process only reinstalled the software that was currently supposed to be running on the machines, then the possibility of infection by malicious code would persist. Instead, the voting machine software software should be modified to defend against installation and viral spreading of unauthorized code. We discuss in Section 5 what software changes are possible and which attacks can be prevented.

3 Design and Operation of the Machine

Before presenting the demonstration attacks we implemented, we will first describe the design and operation of the AccuVote-TS machine and point out design choices that have led to vulnerabilities.

3.1 Hardware

The machine (shown in Figure 1) interacts with the user via an integrated touchscreen LCD display. It authenticates voters and election officials using a motorized smart card reader, which pulls in cards after they are inserted and ejects them when commanded by software. On the right side of the machine is a headphone jack and keypad port for use by voters with disabilities, and a small metal door with a lightweight lock of a variety commonly used in desk drawers and file cabinets. Behind this door is the machine's power switch, a keyboard port, and two PC Card slots, one containing a removable flash memory card and the other optionally containing a modem card used to transfer ballot definitions and election results. The machine is also equipped with a small thermal roll printer for printing records of initial and final vote tallies.

Internally, the machine's hardware closely resembles that of a laptop PC or a Windows CE hand-held device. The motherboard, shown in Figure 2, includes a 133 MHz SH-3 RISC processor, 32 MB of RAM, and 16 MB of flash storage. The machine's power supply can switch to a built-in rechargeable battery in case power is interrupted.

In normal operation, when the machine is switched on, it loads a small bootloader program from its on-board flash memory. The bootloader loads the operating system—Windows CE 3.0—from flash, and then Windows starts the Diebold BallotStation application, which runs the election. Unfortunately, the design allows an attacker with physical access to the inside of the machine's case to force it to run code of her choice [29].

A set of two switches and two jumpers on the motherboard controls the source of the bootloader code that the machine runs when it starts. On reset, the processor begins executing at address 0xA0000000. The switches and jumpers control which of three storage devices—the on-board flash memory, an EPROM chip in a socket on the board, or a proprietary flash memory module in the "ext flash" slot—is mapped into that address range. A table printed on the board lists the switch and jumper configurations for selecting these devices. The capability to boot from a removable EPROM or flash module is useful for initializing the on-board flash when the machine is new or for restoring the on-board flash's state if it gets corrupted, but, as we discussed in Section 2, it could also be used by an attacker to install malicious code.

When we received the machine, the EPROM socket was

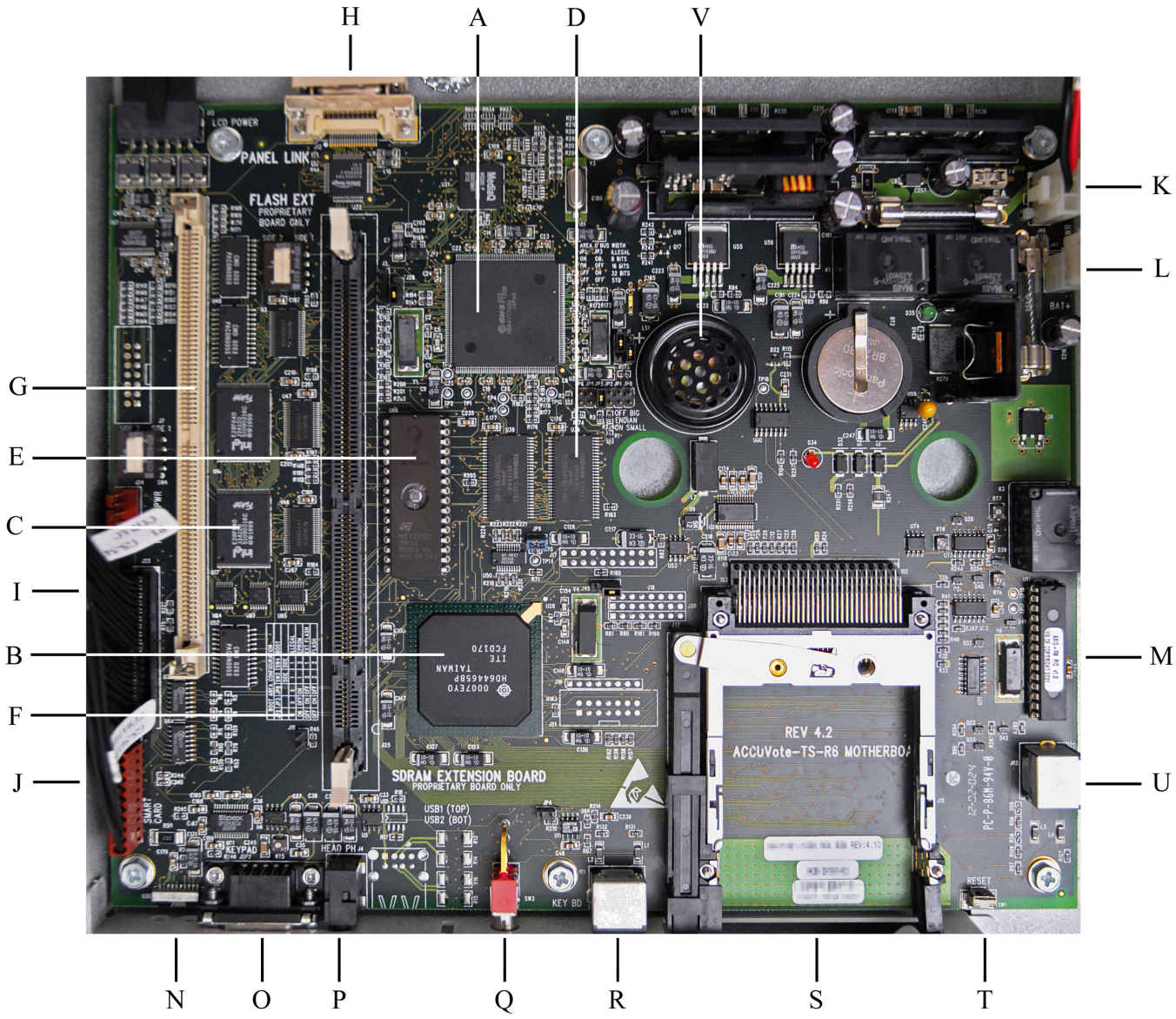

Figure 2: The AccuVote-TS motherboard incorporates a (A) Hitachi SuperH SH7709A 133 MHz RISC microprocessor, (B) Hitachi HD64465 Windows CE Intelligent Peripheral Controller, two (C) Intel Strata-Flash 28F640 8 MB flash memory chips, two (D) Toshiba TC59SM716FT 16 MB SDRAM chips, and a socketed (E) M27C1001 128 KB erasable programmable read-only memory (EPROM). A (F) printed table lists jumper settings for selecting the boot device from among the EPROM, on-board flash, or "ext flash," presumably an external memory inserted in the (G) "Flash Ext" slot.

Connectors on the motherboard attach to the (H) touch sensitive LCD panel, (I) thermal roll printer, and (J) SecureTech ST-20F Smart Card reader/writer, and receive power from the (K) power supply and (L) battery, which are managed by a (M) PIC microcontroller. An (N) IrDA transmitter and receiver, (O) serial keypad connector, and (P) headphone jack are accessible through holes in the machine's case. A (Q) power switch, (R) PS/2 keyboard port, and two (S) PC Card slots can be reached by opening a locked metal door, while a (T) reset switch and (U) PS/2 mouse port are not exposed at all. An (V) internal speaker is audible through the case.

occupied by a 128 KB EPROM containing a bootloader that was older than, but similar to, the bootloader located in the on-board flash. The bootloader contained in the EPROM displays a build date of June 22, 2001 whereas the bootloader contained in the on-board flash displays June 7, 2002. The machine came configured to boot using the on-board flash memory. On our machine, the on-board flash memory is divided into three areas: a 128 KB bootloader, a 3.3 MB GZIP-ed operating system image, and a 10 MB file system partition.

3.2 Boot Process

When the machine is booted, the bootloader copies itself to RAM and initializes the hardware. Then it looks for a memory card in the first PC Card slot, and if one is present, it searches for files on the card with special names. If it finds a file called `fboot.nb0`, it assumes that this file contains a replacement bootloader, and it copies the contents of this file to the bootloader area of the on-board flash memory, overwriting the current bootloader. If it finds a file called `nk.bin`, it assumes that this file contains a replacement operating system image in Windows CE Binary Image Data Format [27], and it copies it to the OS area of the on-board flash, overwriting the current OS image. Finally, if it finds a file called `EraseFFX.bsq`, it erases the entire file system area of the flash. The bootloader does not verify the authenticity of any of these files in any way, nor does it ask the user to confirm any of the changes. As Hursti [18] suggests, these mechanisms can be used to install malicious code.

If none of these files are present, the bootloader proceeds to uncompress the operating system image stored in on-board flash and copy it to RAM, then it jumps to the entry point of the operating system kernel. The operating system image is a kind of archive file that contains an entire Windows CE 3.0 installation, including the kernel's code, the contents of the `Windows` directory, the initial contents of the Windows registry, and information about how to configure the machine's file system.

When Windows starts, the kernel runs the process `Filesys.exe`, which in turn unpacks the registry and runs the programs listed in the `HKEY_LOCAL_MACHINE\Init` registry key [26]. On our machine, these programs are the Debug Shell `shell.exe`, the Device Manager `device.exe`, the Graphics, Windowing, and Events Subsystem `gwes.exe`, and the Task Manager `taskman.exe`. This appears to be a standard registry configuration [25].

The Device Manager is responsible for mounting the file systems. The 10MB file system partition on the on-board flash is mounted at `\FFX`. This partition appears to use the FlashFX file system, a proprietary file system from Datalight, Inc [8]. The memory card, if it is present, is mounted at `\Storage Card`, and may use the FAT or FAT32 file system. The root file system, mounted at `\`, is stored in RAM rather than nonvolatile memory, which causes any files written to it to disappear when the machine is rebooted or otherwise loses power. This design could be leveraged by an attacker who wished to use the file system for temporarily storing data or malicious code without leaving evidence of these activities.

Diebold has customized `taskman.exe` so that it automatically launches the BallotStation application, `\FFX\Bin\BallotStation.exe`. Another customization causes `taskman.exe` to behave differently depending on the contents of any memory cards in the PC Card slots. If a memory card containing a file called `explorer.glb` is present at start-up, `taskman.exe` will invoke Windows Explorer instead of BallotStation. Windows Explorer would give an attacker access to the Windows Start menu, control panels, and file system, as on an ordinary Windows CE machine. The, `taskman.exe` process also searches the memory card for files with names ending in `.ins` [18]. These files are simple scripts in a Diebold-proprietary binary format that automate the process of updating and copying files. Like the special files that the bootloader recognizes, `taskman.exe` accepts `explorer.glb` without authentication of any kind. While `taskman.exe` requests confirmation from the user before running each `.ins` script, we found multiple stack-based buffer overflows in its handling of these files. This suggests that a malformed `.ins` file might be able to bypass the confirmation and cause the machine to execute malicious code.

3.3 Software and Election Procedures

All of the machine's voting-related functions are implemented by BallotStation, a user-space Windows CE application. BallotStation operates in one of four modes: Pre-Download, Pre-Election Testing, Election, and Post-Election. Each corresponds to a different phase of the election process. Here we describe the software's operation under typical election procedures. Our understanding of election procedures is drawn from a number of sources [34, 13, 36, 40] and discussions with election workers from several states. Actual procedures vary somewhat from place to place, and many polling places add additional steps to deal with multiple voter populations (e.g., different parties or electoral districts) and other complicating factors. We omit these details in our description, but we have considered them in our analysis and, except where noted below, they do not affect the results.

At any given time, the machine's mode is determined by the contents of the currently-inserted memory card. Specifically, the current election mode is stored in the header of the election results file, `\Storage`

`Card\CurrentElection\election.brs`. When one memory card is removed and another is inserted, the machine immediately transitions to the mode specified by the card. In addition, if the machine is rebooted, when BallotStation restarts it will return to the mode specified by the current card. As a result, if a machine is powered off while an election is taking place, it will return to Election mode when it is turned back on.

3.3.1 Election Setup

Typically, the voting machines are stored by the local government or the voting machine vendor in a facility with some degree of access control. Before the election (sometimes the night before, or in other cases the same morning) the machines are delivered to polling places where they are set up and prepared by poll workers. Prior to the election, poll workers may configure BallotStation by inserting a memory card containing a ballot description—essentially, a list of races and candidates for the current election. If, instead, a card containing no recognizable election data is inserted into the machine, BallotStation enters Pre-Download mode. In this mode, the machine can download a ballot definition by connecting to a Windows PC running Diebold's GEMS server software.

After election definitions have been installed, BallotStation enters Pre-Election Testing mode. Among other functions, Pre-Election Testing mode allows poll workers to perform so-called "logic and accuracy" (L&A) testing. During L&A testing, poll workers put the machine into a simulation mode where they can cast several test votes and then tally them, checking that the tally is correct. These votes are not counted in the actual election.

After any L&A testing is complete, the poll workers put the machine into Election mode. The software prints a "zero tape" which tallies the votes cast so far. Since no votes have been cast, all tallies should be zero. Poll workers check that this is the case and then sign the zero tape and save it.

3.3.2 Voting

When a voter arrives at the polling place, she checks in at the front desk, where poll workers give her a "voter card," a special smart card that signifies that she is entitled to cast a vote.[7] The voter inserts her voter card into a voting machine, which validates the card. The machine then presents a user interface that allows the voter to express her vote by selecting candidates and answering questions. After making and confirming her selections, the voter pushes a button on the user interface to cast her vote. The machine modifies the voter card, marking it as invalid, and then ejects it. After leaving the machine, the voter returns the now-invalid voter card to the poll workers, who may re-enable it for use by another voter.

3.3.3 Post-Election Activities

At the end of the election, poll workers insert an "Ender Card" to tell the voting software to stop the election and enter Post-Election Mode.[8] Poll workers can then use the machine to print a "result tape" showing the final vote tallies. The poll workers check that the total number of votes cast is consistent with the number of voters who checked in at the front desk. Assuming no discrepancy, the poll workers sign the result tape and save it.

After the result tape is printed, the election results are transferred to the central tabulator, a PC running the GEMS software. Like the ballot definitions, the election results may be transferred over a local area network, a phone line, or a serial cable. Once results from all machines have reached the central tabulator, the tabulator can add up the votes and report a result for the election.

For convenience, it is also possible to "accumulate" the results from several machines into a single AccuVote-TS voting machine, which can then transmit the accumulated results to the central tabulator in a single step. To accumulate results, one machine is put into accumulator mode, and then the memory cards from other machines are inserted (in sequence) into the accumulator machine, which reads the election results and combines them into a single file that will be transferred to the central tabulator or used as an input to further accumulation steps.

If a recount is ordered, the result tapes are rechecked for consistency with voter check-in data, the result tapes are checked for consistency with the results stored on the memory cards, and the tabulator is used again to sum up the results on the memory cards. Further investigation may examine the state stored on memory cards and a machine's on-board file system, such as the machine's logs, to look for problems or inconsistencies.

4 Implementing Demonstration Attacks

To confirm our understanding of the vulnerabilities in the Diebold AccuVote-TS system, and to demonstrate the severity of the attacks that they allow, we constructed demonstration implementations of several of the attacks described above and tested them on the machine. We are not releasing the software code for our demonstration attacks to the public at present; however, a video showing

[7] Kohno *et al.* found numerous vulnerabilities and design flaws in BallotStation's smart card authentication scheme [22], which remain uncorrected in the machine we studied.

[8] They can also use a "Supervisor Card" for this purpose. Supervisor cards enable access to extra setup and administrative operations in pre- and post-election modes.

some of our demonstration attacks in operation is available online at http://itpolicy.princeton.edu/voting.

4.1 Backup and Restore

As a prerequisite to further testing, we developed a method for backing up and restoring the complete contents of the machine's on-board flash memory. This allowed us to perform experiments and develop other demonstration attacks without worrying about rendering the machine inoperable, and it ensured that we could later restore the machine to its initial state for further testing and demonstrations.

We began by extracting the EPROM chip from its socket on the motherboard and reading its 128 KB contents with a universal EPROM programmer. We then disassembled the bootloader contained on the chip using IDA Pro Advanced [9], which supports the SH-3 instruction set. Next, we created a patched version of the EPROM bootloader that searches any memory card[9] in the first PC Card slot for files named `backup.cmd` and `flash.img`. If it finds a file named `backup.cmd`, it writes the contents of the on-board flash to the first 16 MB of the memory card, and if it finds a file named `flash.img`, it replaces the contents of the on-board flash with the contents of that file. We programmed our modified bootloader into a new, standard, 128 KB EPROM chip and inserted it into the motherboard in place of the original chip. We configured the machine to boot using the code in the chip instead of the normal bootloader in its on-board flash memory, as described in Section 3.

4.2 Stealing Votes

Several of the demonstration attacks that we have implemented involve installing code onto AccuVote-TS machines that changes votes so that, for a given race, a favored candidate receives a specified percentage of the votes cast on each affected machine. Since any attacks that significantly alter the total number of votes cast can be detected by election officials, our demonstration software steals votes at random from other candidates in the same race and gives them to the favored candidate. The software switches enough votes to ensure that the favored candidate receives at least the desired percentage of the votes cast on each compromised voting machine.

Election results (i.e., the record of votes cast) are stored in files that can be modified by any program running on the voting machine. The primary copy of the election results is stored on the memory card at `\Storage Card\CurrentElection\election.brs` and a backup copy is stored in the machine's on-board flash memory at `\FFX\AccuVote-TS \BallotStation\CurrentElection\election.brs`. Our software modifies both of these files.

Our demonstration vote-stealing software is implemented as a user-space Windows CE application written in C++ that runs alongside Diebold's BallotStation application. Since our software runs invisibly in the background, ordinary users of BallotStation would not notice its presence. It is pre-programmed with three parameters hard-coded into the binary: the name of the race to rig, the name of the candidate who is supposed to win, and the minimum percentage of the vote that that candidate is to receive.

Alternatively, an attacker could create a graphical user interface that allows more immediate, interactive control over how votes would be stolen. We have also created a demonstration of this kind of attack. In practice, a real attacker would more likely design a vote-stealing program that functioned invisibly, without a user interface.

Our demonstration vote-stealing applications can be generalized to steal votes on behalf of a particular party rather than a fixed candidate, to steal votes only in certain elections or only at certain dates or times, to steal votes only or preferentially from certain parties or candidates, to steal a fixed fraction of votes rather than trying to ensure a fixed percentage result, to randomize the percentage of votes stolen, and so on. Even if the attacker knows nothing about the candidates or parties, he may know that he wants to reduce the influence of voters in certain places. He can do this by creating malicious code that randomly switches a percentage of the votes, and installing that code only in those places. Any desired algorithm can be used to determine which votes to steal and to which candidate or candidates to transfer the stolen votes.

Every time a new memory card is inserted into the machine, our demonstration vote-stealing software looks for an election definition file on the card located at `\Storage Card\CurrentElection\election.edb` and, if one is present, determines whether the current election contains a race it is supposed to rig. If no such race is found, the software continues to wait. If a target race is found, it searches that race for the name of the favored candidate. Upon finding that the preferred candidate is on the ballot, the software proceeds to poll the election result files every 15 seconds to see if they have been changed.

If the demonstration vote-stealing software successfully opens the result files during one of its polling attempts, it first checks the result files' headers to see whether the machine is in Election mode. If not, the attack software does not change any votes. This feature ensures that the software would not be detected during Logic and Accuracy testing, which occurs when the machine is in Pre-Election Testing mode. The software could be further enhanced so that it would only change votes during a specified period

[9] While Diebold sells special-purpose memory cards for use in the machine, we were able to substitute a CompactFlash card (typically used in digital cameras) and a CompactFlash-to-PC Card adapter.

on election day, or so that it would only change votes in the presence or absence of a "secret knock." A secret knock is a distinctive sequence of actions, such as touching certain places on the screen, that an attacker executes in order to signal malicious software to activate or deactivate itself.

If the machine is in election mode and the demonstration vote-stealing software successfully opens the result files, then the software checks whether any new ballots have been cast since the last time it polled the files. For each new ballot cast, the software determines whether the race being rigged is on that ballot, and if so, determines whether the corresponding result record contains a vote for the favored candidate or for an opponent. The software maintains a data structure that keeps track of the location of every result record that contains a vote for an opponent of the favored candidate so that it can come back later and change some of those records if necessary. Since each result record is only labeled with the ID number of the ballot to which it corresponds, the software must look up each record's ballot ID in the election definition file in order to determine which candidates the votes in the record are for.

Once it has parsed any newly cast ballots, the software switches the minimum number of votes necessary to ensure that the favored candidate gets at least the desired percentage of the vote. The vote-stealing software chooses which votes to switch by selecting entries at random from its data structure that tracks votes for the opponents of the favored candidate. After the necessary changes have been made to the result files, the software closes the files, resumes the BallotStation process, and continues to wait in the background.

The steps described above are all that is necessary to alter every electronic record of the voters' intent that an AccuVote-TS machine produces. Several of the machine's supposed security features do not impede this attack. The so-called "protective counter," supposedly an unalterable count of the total number of ballots ever cast on the machine, is irrelevant to this attack because the vote-stealing software does not change the vote count.[10] The machine's audit logs are equally irrelevant to this attack because the only record they contain of each ballot cast is the log message "Ballot cast." Furthermore, the fact that election results are stored redundantly in two locations is not an impediment because the vote-stealing software can modify both copies. Finally, as discussed in Section 2, the fact that the election results are encrypted does not foil this attack.

[10] In any event, the "protective counter" is simply an integer stored in an ordinary file, so an attack that needed to modify it could do so easily [22].

4.3 Demonstration Voting Machine Virus

In addition to our demonstration vote-stealing attacks, we have developed a voting machine virus that spreads the vote-stealing code automatically and silently from machine to machine. The virus propagates via the removable memory cards that are used to store the election definition files and election results, and for delivering firmware updates to the machines. It exploits the fact, discovered by Hursti [18], that when the machine boots, the Diebold bootloader will install any code found on the removable memory card in a file with the special name `fboot.nb0`. As a result, an attacker could infect a large population of machines while only having temporary physical access to a single machine or memory card.

Our demonstration virus takes the form of a malicious bootloader that infects a host voting machine by replacing the existing bootloader in the machine's on-board flash memory. Once installed, the virus deploys our demonstration vote-stealing software and copies itself to every memory card that is inserted into the infected machine. If those cards are inserted into other machines, those machines can become infected as well.

The cycle of infection proceeds as follows. When the virus is carried on a memory card, it resides in a 128 KB bootloader image file named `fboot.nb0`. This file contains both the malicious replacement bootloader code and a Windows CE executable application that implements the demonstration vote-stealing application. The vote-stealing executable is stored in a 50 KB region of the bootloader file that would normally be unused and filled with zeroes.

When a card carrying the virus is inserted into a voting machine and the machine is switched on or rebooted, the machine's existing bootloader interprets the `fboot.nb0` file as a bootloader update and copies the contents of the file into its on-board flash memory, replacing the existing bootloader with the malicious one. The original bootloader does not ask for confirmation before replacing itself. It does display a brief status message, but this is interspersed with other normal messages displayed during boot. These messages are visible for less than 20 seconds and are displayed in small print at a 90 degree angle to the viewer. After the boot messages disappear, nothing out of the ordinary ever appears on the screen.

Once a newly infected host is rebooted, the virus bootloader is in control. Since the bootloader is the first code that runs on the machine, a virus bootloader is in a position to affect all aspects of system operation. While booting, the virus bootloader, like the ordinary bootloader, checks for the presence of a memory card in the first PC Card slot. However, if it finds a bootloader software update on the card, it pretends to perform a bootloader update by printing out the appropriate messages, but actually does

nothing.[11] Thus, once a machine has been infected, the only way to remove the virus bootloader is to restore the machine's state using an EPROM-resident bootloader.

If a memory card is present, the virus bootloader copies itself to the card as a file named `fboot.nb0` so that it can spread to other machines. If the card already contains a file with that name, the bootloader replaces it. Consequently, if a service technician performing bootloader updates tries to update an infected machine using a card containing an `fboot.nb0` file, the infected machine will not be updated (although it will pretend to be), and all subsequent machines that the technician tries to update using the same card will receive the virus bootloader instead of the updated one. Similarly, updates to the BallotStation software or operating system can also propagate the virus.

The malicious bootloader also copies the vote-stealing executable to the memory card as a file named `AV.EXE`. Then, immediately before starting Windows, the virus bootloader scans the region of RAM occupied by the operating system image (0x8C080000–0x8C67FFFF) for the hard-coded string in the `taskman.exe` binary that points to the BallotStation executable `\FFX\Bin\BallotStation.exe` and replaces it with `\Storage Card\AV.EXE`. Consequently, when Windows starts, `taskman.exe` will launch the demonstration vote-stealing application instead of BallotStation.

When the demonstration vote-stealing application on the memory card starts, it first renames the legitimate BallotStation executable to `\FFX\Bin\AccuVote.exe`, and then it copies itself to the machine's on-board flash memory with the name `\FFX\Bin\BallotStation.exe`. It adopts the name of the BallotStation executable so it will still run at start-up even if the machine is booted without a memory card in the first PC Card slot. Next, it copies the malicious bootloader image from the card to the on-board flash . Thereafter, the software periodically checks whether an uninfected memory card is present in the machine, and, if so, it copies the virus files onto the card so that other machines where the card is used will become infected. Finally, the vote-stealing application runs in the background, changing votes in the manner described in Section 4.2.

[11] In order to avoid printing out fake update messages when the copy of `fboot.nb0` on the card was put there by the virus bootloader itself, whenever the virus bootloader copies itself to a card, it sets the hidden, system, and read-only FAT attributes of the resulting `fboot.nb0` file. Then, when the virus bootloader checks for the presence of the `fboot.nb0` file on the card, it only prints out fake update messages if the file does not have those attributes. Alternatively, the virus bootloader could identify copies of itself by examining the contents of the `fboot.nb0` file for some characteristic bit string.

4.4 Demo Denial-of-Service Attack

To illustrate how malicious software running on an AccuVote-TS could launch a denial-of-service (DoS) attack, we developed a demonstration attack program that, on command, erases the contents of both the currently-inserted memory card and the machine's on-board flash memory. This attack not only destroys all records of the election currently in progress (both the primary and backup copies), but also renders the machine inoperable until a service technician has the opportunity to dismantle it and restore its configuration.

The demonstration DoS program is comprised of a user-space Windows CE executable that triggers the attack and a malicious bootloader that functions like an ordinary bootloader, except that upon receiving the appropriate signal, it completely erases the currently-inserted memory card and the machine's on-board flash memory. The user-space trigger program works by first writing a special value to a part of the machine's on-board flash memory that is accessible from user-space programs and then crashing the machine by invoking the `PowerOffSystem()` Windows CE API call. The `PowerOffSystem()` API is supposed to put the system in a low-power "sleep" mode from which it can later "wake-up," but when this API is invoked on an AccuVote-TS, the machine simply crashes. When the machine is rebooted (which must be done manually), the malicious bootloader notices that the special value has been written to the machine's on-board flash memory. On this signal, it completely erases the contents of both the currently-inserted memory card and the machine's on-board flash memory. In so doing, the malicious bootloader destroys all of the data, software, and file system formatting on both the memory card and the on-board flash memory.

In order to account for the possibility that the malicious bootloader never gets a chance to completely erase both storage media or that the memory card is removed before the machine is rebooted, the user-space trigger program deletes as much as it can before crashing the machine. It deletes all of the files on the memory card and on the machine's on-board `\FFX` file system including both the primary and backup copies of the election results (`election.brs`), the audit logs (`election.adt`), and the BallotStation executable. When it deletes these files, it overwrites each of them with garbage data to make it less likely that the files will ever be recovered.

While our demonstration DoS attack is triggered by a user's command, a real attacker could create malicious software that only triggers the above attack on a specific date and time, such as on election day. An attacker could also design the attack to launch in response to a specific trend in voting during an election, such as an apparent victory for a particular candidate. Like a vote-stealing

attack, a DoS attack could be spread by a virus.

5 Mitigation

The vulnerabilities that we have described can be mitigated, to some extent, by changing voting machine designs and election procedures. In this section we discuss several mitigation strategies and their limitations.

5.1 Software and Hardware Modifications

The AccuVote-TS machine is vulnerable to computer viruses because it automatically loads and runs code found on memory cards without authenticating it. Its software could be redesigned to inhibit the spread of viruses, however. One approach would be to digitally sign all software updates and have the machine's software verify the signature of each update before installing it. Such a change would ensure that only updates signed by the manufacturer or another trusted certifying authority could be loaded.[12] It would also be helpful to require the person using the machine to confirm any software updates. Confirmation of updates would not prevent a malicious person with physical access to the machine from loading an update, but at least it would make the accidental spread of a virus less likely while the machine was being used by honest election officials.

While redesigning the voting machine's software can help mitigate some of the security problems that we identify, there are other problems inherent in the AccuVote-TS hardware architecture that cannot be addressed by software changes. For example, there is nothing to stop an adversary who has physical access to the machine from booting and installing his own malicious software by replacing the socketed EPROM chip on the motherboard. Furthermore, because all of the machine's state is kept in rewritable storage (RAM, flash memory, or a memory card), it is impossible to create tamper-proof logs, records, and counters. In addition, as is the case with ordinary PCs, it is difficult to determine with certainty that the machine is actually running the software that it is supposed to run. Rootkit techniques [16] and virtualization technologies [21], which are often used to conceal malware in the PC setting, could be adapted for use on the voting machines.

Researchers have proposed various strategies for building specialized hardware capable of maintaining tamper-proof and tamper-evident logs, records, and counters (e.g., [37]), as well as software strategies that provide more limited protection (e.g., [33]). Although such methods could prevent attacks that aim to alter votes after they have been recorded, they could not prevent malicious code from changing future votes by altering data before it is sent to the storage device.

Assuring a computer's software configuration is also a notoriously difficult problem, and research has focused on mechanisms to ensure that only approved code can boot [1] or that a machine can prove to a remote observer that it is running certain code [37]. For example, commercial systems such as Microsoft's Xbox game console have incorporated mechanisms to try to resist modification of the boot code or operating system, but they have not been entirely successful [17]. Although mechanisms of this type are imperfect and remain subjects of active research, they seem appropriate for voting machines because they offer some level of assurance against malicious code injection. It is somewhat discouraging to see voting machine designers spend much less effort on this issue than game console designers.

While changes to the hardware and software of voting machines can reduce the threats of malicious code injection and log tampering, no purely technical solution can eliminate these problems.

5.2 Physical Access Controls

Despite the best efforts of hardware and software designers, any physical access to a computer still raises the possibility of malicious code installation, so election officials should limit access to voting machines' internals, their memory cards, and their memory card slots to the extent possible.

There is some benefit in sealing the machine's case, memory card, and card bay door with individually numbered tamper-evident seals, in the hope of detecting illicit accesses to these areas. While these measures may expose some classes of attacks, they make denial-of-service attacks easier. Suppose, for example, that a malicious voter cuts a seal while an election is in progress. If machines with broken seals are treated as completely untrustworthy, then cutting the seal is itself an effective denial-of-service attack. If broken seals are usually ignored when everything else seems to be in order, then an attacker has a good chance of successfully inserting malicious code that cleans up after itself. There seems to be no fully satisfactory compromise point between these two extremes.

Even leaving aside the possibility that voters will deliberately break seals, broken seals are an unfortunately common occurrence. The most comprehensive study of AccuVote DRE election processes in practice examined the May 2006 primary election in Cuyahoga County, Ohio, which used AccuVote-TSx machines. The study found that more than 15% of polling places reported at least one

[12] Adding signatures would not be effective if a machine has already been infected with malicious code; machines would need to be booted from EPROM and completely restored to a known state before their software were updated to a version that checked signatures.

problem with seals [13].

The available evidence is that machines and memory cards are not handled with anything approaching the necessary level of care. For example, the Cuyahoga County study [13] reported many procedural weaknesses: "A lack of inventory control and gaps in the chain of custody of mission critical assets (i.e. DRE memory cards, [DREs], ...)" (p. 103); "the systems of seals, signatures and other security features of the... machine memory cards were not implemented on a consistent basis" (p. 109); "It appears that memory cards are regularly removed and re-inserted when a DRE becomes out-of-service. Security tabs are broken and no log of this remove and replace activity is maintained... There is no indication that a record comparing memory card to DRE serial number is kept" (p. 138); "Security seals are not checked for integrity at the end of Election Day, nor are they matched with a deployment list of Security seal serial numbers. There is no attempt to reconcile memory cards intended for the precinct with memory cards removed from the DREs at the end of the day... Therefore, it is unknown whether these memory cards were tampered with during Election Day" (p. 139); "There is no established chain of custody during the transfer of the memory cards... from the vote center to the BOE [Board of Elections]" (p. 140); "Security seals are collected upon return to the BOE, but these serial numbers are neither logged nor checked against the original security seal serial numbers deployed with the memory cards. Therefore, it is unknown whether these memory cards were tampered with during transport to the BOE from the polling location" (p. 140). These problems require immediate attention from election officials.

Security seals do some good, but it is not a solution simply to assume that seals will always be used, always be checked, and never be broken. Inevitably, some seals will be missing or broken without an explanation, providing potential cover for the insertion of malicious code or a voting machine virus.

5.3 Effective Parallel Testing

In parallel testing, election officials choose some voting machines at random and set them aside, casting simulated votes on them throughout election day and verifying at the end of the election that the machines counted the simulated votes correctly. The goal of parallel testing is to trigger and detect any vote-stealing software that may be installed on the machines.

A challenge in parallel testing is how to make the simulated voting pattern realistic. If the pattern is unrealistic in some respect—if, say, the distribution of votes throughout the day doesn't match what a real voting machine would see—then vote-stealing software may be able to tell the difference between a real election and parallel testing, allowing the software to steal votes in the real election while leaving results unchanged in parallel testing.

Parallel testing is also vulnerable to a "secret knock" attack by a testing insider. Generally, parallel tests are carried out by representatives from all political parties to ensure impartiality. However, if one representative has placed vote altering code on the machines, she could disable the code on the machine being tested by issuing a surreptitious command. For example, the code might watch for a specific sequence of touches in a normally unused area of the screen and deactivate its vote altering function in response. Preventing this kind of attack requires carefully scripting the testing procedure.

Alternatively, a secret knock might be used to activate malicious code. In this scheme, malicious voters would perform the secret knock on the machines being used to collect real votes, or a malicious election worker would perform it surreptitiously when setting up the machines, and vote-stealing software would wait for this secret knock before operating. Machines chosen for parallel testing would not see the secret knock and so would count votes honestly. This approach has the drawback (for the attacker) of requiring a significant number of malicious voters or a malicious poll worker to participate, though these participants would not have to know all the details of the attack.

These possibilities reduce the usefulness of parallel testing in practice, but we think it can still be a worthwhile precaution when conducted according to rigorously controlled procedures.

5.4 Effective Whole-System Certification

Despite their very serious security flaws, the Diebold DREs were certified according to federal and state standards. This demonstrates that the certification processes are deficient. The Federal Election Commission's 2002 Voting System Standards [14] say relatively little about security, seeming to focus instead on the machine's reliability if used non-maliciously.

The U.S. Election Assistance Commission issued voluntary voting system guidelines [38] in 2005. These are considerably more detailed, especially in the area of security, than the FEC's 2002 standards. Though it would not be entirely fair to apply the 2005 guidelines to the pre-2005 version of the AccuVote software we studied, we do note that the AccuVote-TS hardware architecture may make it impossible to comply with the 2005 guidelines, in particular with the requirement to detect unauthorized modifications to the system software (see [38], Volume I, Section 7.4.6). In practice, a technology can be deployed despite noncompliance with certification requirements if the testing agencies fail to notice the problem.

In general, the certification process seems to rely more

on testing than analysis. Testing is appropriate for some properties of interest, such as reliability in the face of heat, cold, and vibration, but testing is ill-suited for finding security problems. As discussed frequently in the literature, testing can only show that a system works under specific, predefined conditions; it generally cannot ensure that there is no way for an attacker to achieve some goal by violating these conditions. Only a competent and thorough security analysis can provide any confidence that the system can resist the full range of realistic attacks.

Weak certification would be less of a problem if information about the system's design were more widely available to the public. Researchers and other experts would be able to provide valuable feedback on voting machine designs if they had the information to do so. Ideally, strong certification procedures would be coupled with public scrutiny to provide the highest assurance.

5.5 Software-Independent Design

Although the strategies described above can contribute to the integrity of election results, none are sufficient to mitigate the vote-stealing attacks that we have demonstrated. The only known method of achieving an acceptable level of security against the attacks we describe is software-independent design. "A voting system is *software-independent* if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome [31]." In the near term, the only practical way to make DREs software-independent is through the use of a voter-verifiable paper audit trail (VVPAT) coupled with random audits. The VVPAT creates a paper record, verified visually by the voter, of how each vote was cast. This record can be either a paper ballot that is deposited by the voter in a traditional ballot box, or a ballot-under-glass system that keeps the paper record within the voting machine but lets the voter see it [24]. A VVPAT makes our vote-stealing attack detectable. In an all-electronic system like the Diebold DREs, malicious code can modify all of the logs and records in the machine, thereby covering up its vote stealing, but the machine cannot modify already created paper records, and the accuracy of the paper records is verified by voters.

Paper trails have their own failure modes, of course. If they are poorly implemented, or if voters do not know how or do not bother to check them, they may have little value [3, 13]. The real advantage of a paper trail is that its failure modes differ significantly from those of electronic systems, making the combination of paper and electronic record keeping harder to defraud than either would be alone. Requiring a would-be vote stealer to carry out both a code-injection attack on the voting machines and a physical ballot box stuffing attack would significantly raise the difficulty of attacking the system.

Paper ballots are only an effective safeguard if they are actually used to check the accuracy of the machines. This need not be done everywhere. It is enough to choose a small fraction of the polling places at random and verify that the paper ballots match the electronic records there. If the polling places to recount are chosen by a suitable random procedure, election officials can establish with high probability that a full comparison of paper and electronic records would not change the election's result. Methods for conducting these random audits are discussed by Rivest [2] and Calandrino, *et al.* [6], among others.

Another limitation of VVPATs is that they cannot stop a denial-of-service attack from spoiling an election by disabling a large number of voting machines on election day. Given this possibility, if DREs are used, it is worthwhile to have an alternative voting technology available, such as paper ballots.

In the future, cryptographic voting may provide an alternative means of achieving software-independence that offers greater security than VVPATs. Cryptographic voting systems (e.g., [32, 5]) aim not only to allow voters to verify that their votes were recorded as cast, but also to allow them to confirm that their ballots were actually included in the final vote totals. Currently, however, achieving acceptable usability and maintaining ballot secrecy remain challenges for such schemes (see [19]).

6 Related Work

Several previous studies have criticized the security of the Diebold AccuVote DRE systems. The first major study of these machines was published in 2003 by Kohno *et al.* [22], who did not have access to a machine but did have a leaked version of the source code for BallotStation. They found numerous security flaws in the software and concluded that its design did not show evidence of any sophisticated security thinking. They did not study the AccuVote-TS's kernel or bootloader, however.

Public concern in light of Kohno's study led the state of Maryland to authorize two security studies. The first study, by SAIC, reported that the system was "at high risk of compromise" [34]. RABA, a security consulting firm, was hired to do another independent study of the Diebold machines. RABA had access to a number of machines and some technical documentation. Their study [30] was generally consistent with Kohno's findings, and found some new vulnerabilities. It suggested design changes to the Diebold system, and outlined some steps that Maryland might take to reduce the risk of security problems. The state responded by adopting many of RABA's suggestions [23].

A further security assessment was commissioned by the Ohio Secretary of State and carried out by the Compuware Corporation [7]. This study examined several

DRE systems, including the AccuVote-TS running the same version of BallotStation as our machine, and identified several high risk security problems.

In 2006, in response to reports that Harri Hursti had found flaws in Diebold's AccuBasic subsystem, the state of California asked Wagner, Jefferson, and Bishop to perform a study of AccuBasic security issues. Their report [39] identified several vulnerabilities that differ from those that we describe because the machine that we studied lacks the AccuBasic subsystem.

Later in 2006, Hursti released a report [18] describing several security weaknesses in the AccuVote-TS and -TSx systems that could allow an attacker to install malicious software by subverting the systems' software update mechanisms. These weaknesses form the basis for many of the attacks that we describe in the current study. With limited access to the voting machines, Hursti could only confirm that one of these weaknesses could be exploited; we show that many of the others can be as well.

Our work builds on these previous reports. Our findings generally confirm the behaviors and vulnerabilities described by Kohno *et al.*, RABA, and Hursti, and demonstrate through proof-of-concept implementations that the vulnerabilities can be exploited to implement viral attacks and to change election results. To our knowledge, our work is the first comprehensive, public description of these threats to Diebold's DREs.

Several studies discuss general issues in the construction of software-based attacks on DRE voting machines. Kelsey [20] catalogs the attacker's design choices; our analysis confirms that all or nearly all of the attack options Kelsey discusses can be carried out against the Diebold machine we studied. The Brennan Center report [3] offers a broader but less technical discussion; its discussion of malicious software injection attacks is based partially on Kelsey's analysis.

Additionally, there is an extensive literature on electronic voting in general, which we will not attempt to survey here.

7 Conclusion

From a computer security standpoint, DREs have much in common with desktop PCs. Both suffer from many of the same security and reliability problems, including bugs, crashes, malicious software, and data tampering. Despite years of research and enormous investment, PCs remain vulnerable to these problems, so it is doubtful, unfortunately, that DRE vendors will be able to overcome them.

Nevertheless, the practical question facing public officials is whether DREs provide better security than other election technologies, which have their own history of failure and fraud. DREs may resist small-scale fraud as well as, or better than, older voting technologies; but DREs are much more vulnerable to large-scale fraud. Attacks on DREs can spread virally, they can be injected far in advance and lurk passively until election day, and they can alter logs to cover their tracks. Procedures designed to control small-scale fraud are no longer sufficient—DREs demand new safeguards.

Electronic voting machines have their advantages, but experience with the AccuVote-TS and other paperless DREs shows that they are prone to very serious vulnerabilities. Making them safe, given the limitations of today's technology, will require safeguards beginning with software-independent design and truly independent security evaluation.

Acknowledgments

We thank Andrew Appel, Jeff Dwoskin, Laura Felten, Shirley Gaw, Brie Ilenda, Scott Karlin, Yoshi Kohno, David Robinson, Avi Rubin, Adam Stubblefield, Dan Wallach, and Harlan Yu for technical help, information, and feedback. We are especially grateful to the party who provided us the machine to study.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] ARBAUGH, W., FARBER, D., AND SMITH, J. A secure and reliable bootstrap architecture. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pp. 65–71.

[2] ASLAM, J., POPA, R., AND RIVEST, R. On estimating the size of a statistical audit. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*.

[3] BRENNAN CENTER TASK FORCE ON VOTING SYSTEM SECURITY. *The Machinery of Democracy: Protecting Elections in an Electronic World*. 2006.

[4] BROACHE, A. Diebold reveals 'key' to e-voting? *CNet News.com* (Jan. 2007). Available at http://news.com.com/2061-10796_3-6153328.html.

[5] C. A. NEFF. Practical high certainty intent verification for encrypted votes. Available at http://www.votehere.com/vhti/documentation/vsv-2.0.3638.pdf, 2004.

[6] CALANDRINO, J., HALDERMAN, J. A., AND FELTEN, E. Machine-assisted election auditing. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*.

[7] COMPUWARE. Direct recording electronic (DRE) technical security assessment report. Available at http://www.sos.state.oh.us/sos/hava/compuware112103.pdf, 2003.

[8] DATALIGHT. FlashFX product details. Available at http://datalight.com/products/flashfx/productdetails.php.

[9] DATARESCUE. IDA Pro Disassembler. Available at http://www.datarescue.com/idabase.

[10] DIEBOLD ELECTION SYSTEMS. Checks and balances in elections equipment and procedures prevent alleged fraud scenarios. Available at http://www.votetrustusa.org/pdfs/DieboldFolder/checksandbalances.pdf, 2003.

[11] DIEBOLD ELECTION SYSTEMS. Press release: State of Maryland awards Diebold electronic voting equipment order valued at up to $55.6 million. Available at http://www.diebold.com/news/newsdisp.asp?id=2979, 2003.

[12] ELECTION DATA SERVICES. 2006 voting equipment study. Available at http://www.edssurvey.com/images/File/ve2006_nrpt.pdf.

[13] ELECTION SCIENCE INSTITUTE. DRE analysis for May 2006 primary, Cuyahoga County, Ohio. Available at http://bocc.cuyahogacounty.us/GSC/pdf/esi_cuyahoga_final.pdf, Aug. 2006.

[14] FEDERAL ELECTION COMMISSION. Voting system standards. Available at http://www.eac.gov/election_resources/vss.html, 2002.

[15] HAUG, N. Vendor proposal evaluation findings report and addendum. Available at http://www.sos.state.oh.us/sos/hava/findings091003.pdf, 2003.

[16] HOGLUND, G., AND BUTLER, J. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.

[17] HUANG, A. *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, 2003.

[18] HURSTI, H. Critical security issues with Diebold TSx (unredacted). Available at http://www.bbvdocs.org/reports/BBVreportIIunredacted.pdf, May 2006.

[19] KARLOF, C., SASTRY, N., AND WAGNER, D. Cryptographic voting protocols: A systems perspective. In *Proc. 14th USENIX Security Symposium* (2005).

[20] KELSEY, J. Strategies for software attacks on voting machines. In *NIST Workshop on Threats to Voting Systems* (2005).

[21] KING, S., CHEN, P., WANG, Y., VERBOWSKI, C., WANG, H., AND LORCH, J. SubVirt: Implementing malware with virtual machines. In *Proc. 2006 IEEE Symposium on Security and Privacy*, pp. 314–327.

[22] KOHNO, T., STUBBLEFIELD, A., RUBIN, A., AND WALLACH, D. Analysis of an electronic voting system. In *Proc. 2004 IEEE Symposium on Security and Privacy*, pp. 27–42.

[23] MARYLAND STATE BOARD OF ELECTIONS. Response to: Department of legislative services trusted agent report on Diebold AccuVote-TS voting system. Available at http://mlis.state.md.us/Other/voting_system/sbe_response.pdf, 2004.

[24] MERCURI, R. *Electronic Vote Tabulation: Checks and Balances*. PhD thesis, University of Pennsylvania, 2001.

[25] MICROSOFT. Configuring the process boot phase. Available at http://msdn2.microsoft.com/en-us/library/ms901773.aspx.

[26] MICROSOFT. Filesys.exe boot process. Available at http://msdn2.microsoft.com/en-us/library/ms885423.aspx.

[27] MICROSOFT. Windows CE binary image data format specification. Available at http://msdn2.microsoft.com/en-us/library/ms924510.aspx.

[28] NEUMANN, P. Security criteria for electronic voting. In *16th National Computer Security Conference* (1993).

[29] OPEN VOTING CONSORTIUM. Worst ever security flaw found in Diebold TS voting machine. Available at http://www.openvotingconsortium.org/blog/2006-jul-31/worst_flaw_ever_in_diebold_touch_screen_voting_machine_revealed, 2006.

[30] RABA TECHNOLOGIES. Trusted agent report: Diebold AccuVote-TS voting system. Available at http://www.raba.com/press/TA_Report_AccuVote.pdf, 2004.

[31] RIVEST, R., AND WACK, J. On the notion of "software independence" in voting systems. Available at http://vote.nist.gov/SI-in-voting.pdf, July 2006.

[32] S. POPOVENIUC AND B. HOSP. An introduction to punchscan. Available at http://www.punchscan.org/papers/popoveniuc_hosp_punchscan_introduction.pdf, 2006.

[33] SCHNEIER, B., AND KELSEY, J. Cryptographic support for secure logs on untrusted machines. In *Proc. 7th USENIX Security Symposium* (1998).

[34] SCIENCE APPLICATIONS INTERNATIONAL CORPORATION. Risk assessment report: Diebold AccuVote-TS voting system and processes (unredacted). Available at http://www.bradblog.com/?p=3731, 2003.

[35] SONGINI, M. E-voting security under fire in San Diego lawsuit. *Computerworld* (Aug. 2006).

[36] STATE OF MARYLAND. Code of Maryland regulations, Title 33, State Board of Elections. Available at http://www.dsd.state.md.us/comar/subtitle_chapters/33_Chapters.htm.

[37] TRUSTED COMPUTING GROUP. TCG TPM specification. Available at https://www.trustedcomputinggroup.org/specs/TPM.

[38] UNITED STATES ELECTION ASSISTANCE COMMISSION. Voluntary voting systems guidelines. Available at http://www.eac.gov/vvsg_intro.htm, 2005.

[39] WAGNER, D., JEFFERSON, D., AND BISHOP, M. Security analysis of the Diebold AccuBasic interpreter. Available at http://www.ss.ca.gov/elections/voting_systems/security_analysis_of_the_diebold_accubasic_interpreter.pdf, Feb. 2006.

[40] WILLIAMS, B. Security in the Georgia voting system. Available at http://www.votescount.com/georgia.pdf, 2003.